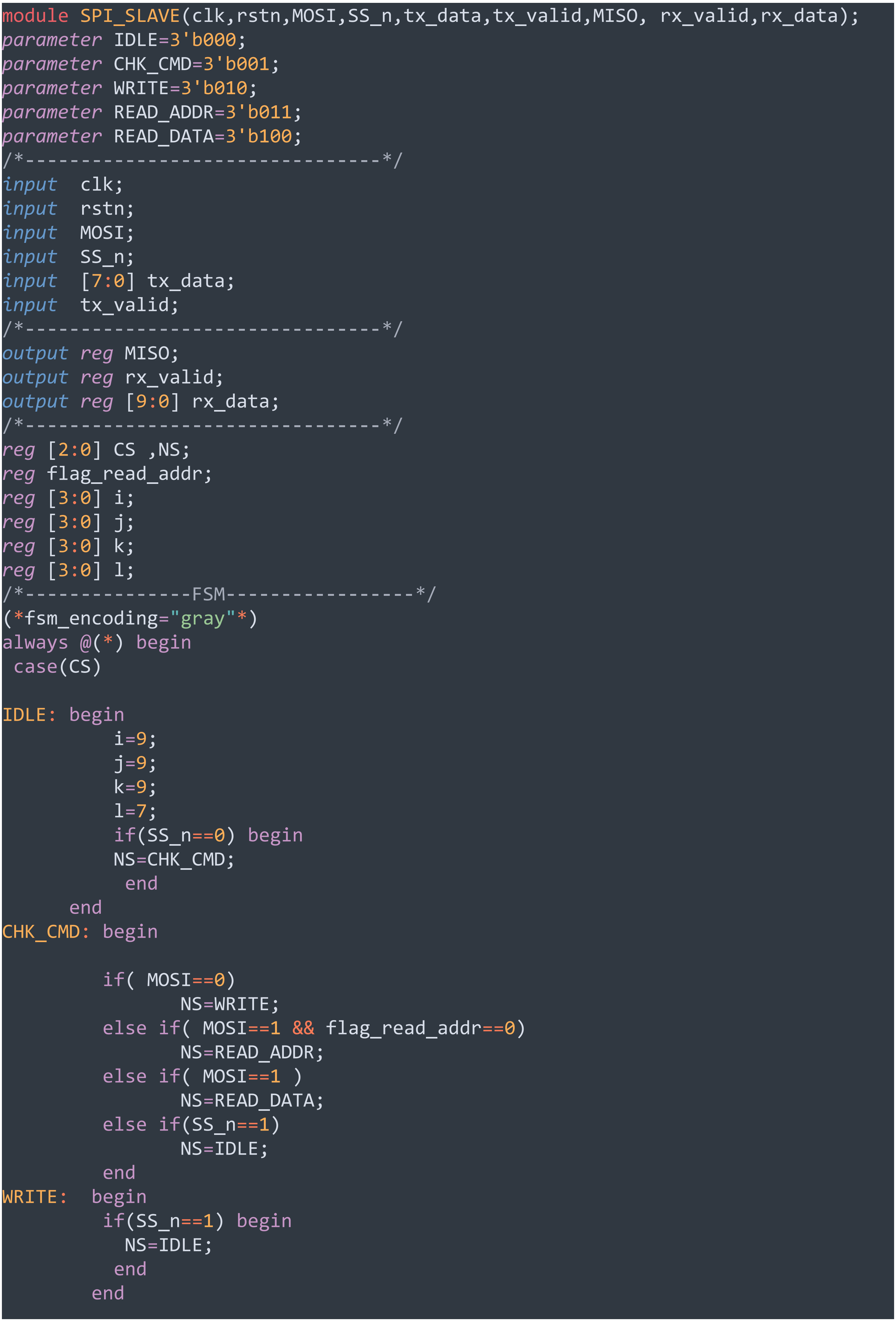
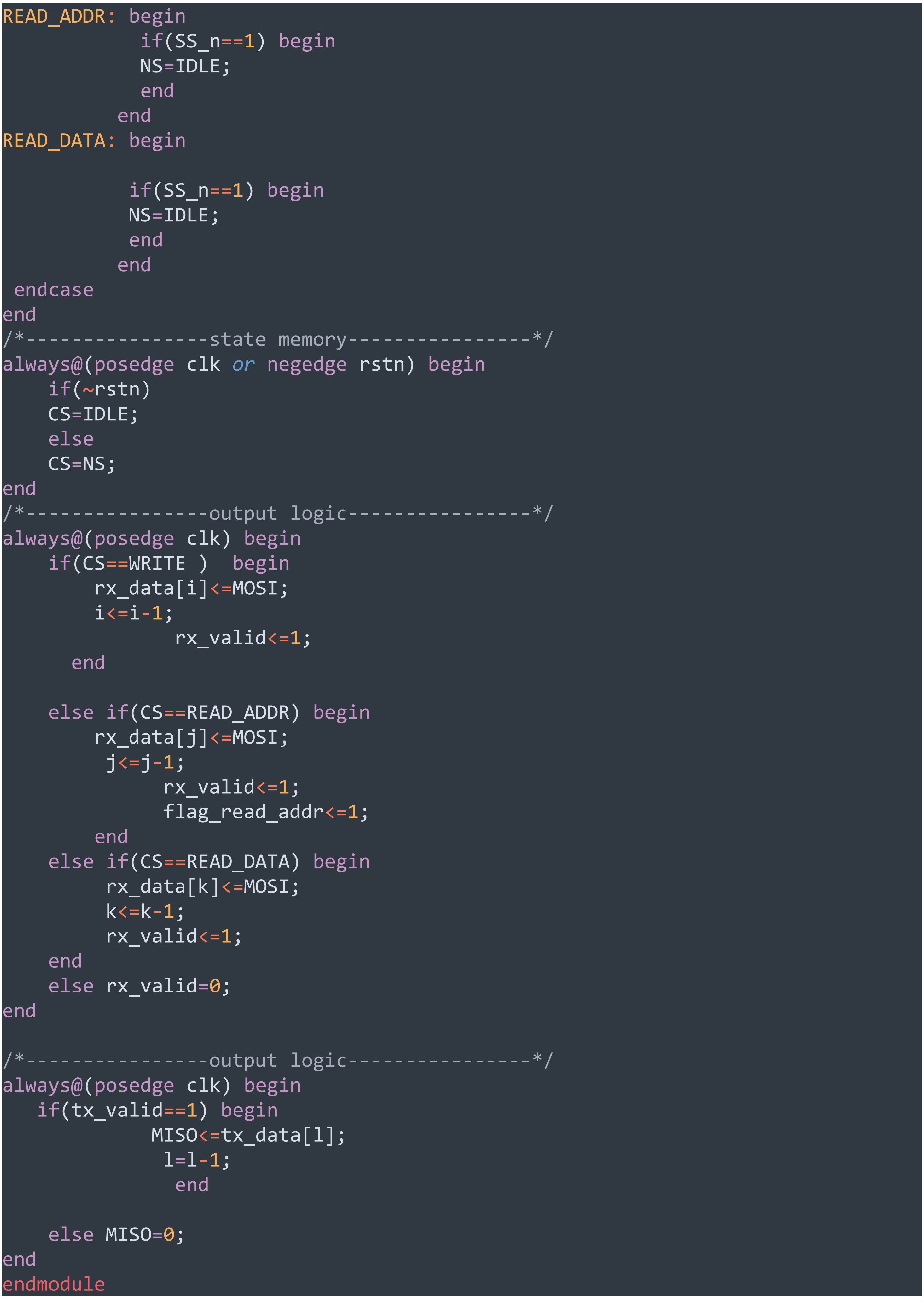
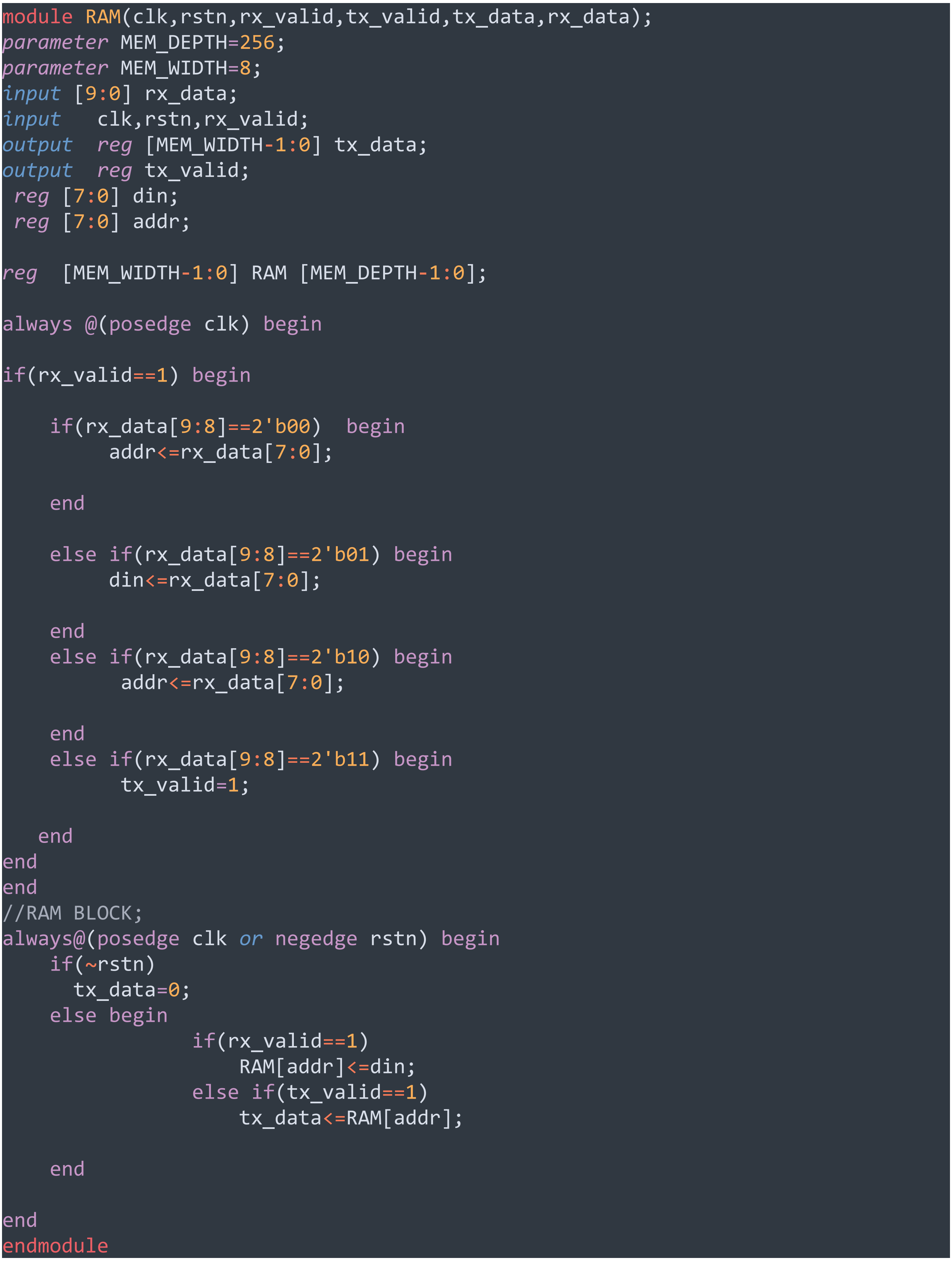


```
module SPI_SLAVE(clk,rstn,MOSI,SS_n,tx_data,tx_valid,MISO, rx_valid,rx_data);
parameter IDLE=3'b000;
parameter CHK_CMD=3'b001;
parameter WRITE=3'b010;
parameter READ_ADDR=3'b011;
parameter READ_DATA=3'b100;
/*--------------------------------*/
input  clk;
input  rstn;
input  MOSI;
input  SS_n;
input  [7:0] tx_data;
input  tx_valid;
/*--------------------------------*/
output reg MISO;
output reg rx_valid;
output reg [9:0] rx_data;
/*--------------------------------*/
reg [2:0] CS ,NS;
reg flag_read_addr;
reg [3:0] i;
reg [3:0] j;
reg [3:0] k;
reg [3:0] l;
/*---------------FSM----------------*/
(*fsm_encoding="gray"*)
always @(*) begin
 case(CS)

IDLE: begin
          i=9;
          j=9;
          k=9;
          l=7;
          if(SS_n==0) begin
          NS=CHK_CMD;
           end
      end
CHK_CMD: begin

         if( MOSI==0)
                NS=WRITE;
         else if( MOSI==1 && flag_read_addr==0)
                NS=READ_ADDR;
         else if( MOSI==1 )
                NS=READ_DATA;
         else if(SS_n==1)
                NS=IDLE;
         end
WRITE:  begin
         if(SS_n==1) begin
           NS=IDLE;
          end
        end
```

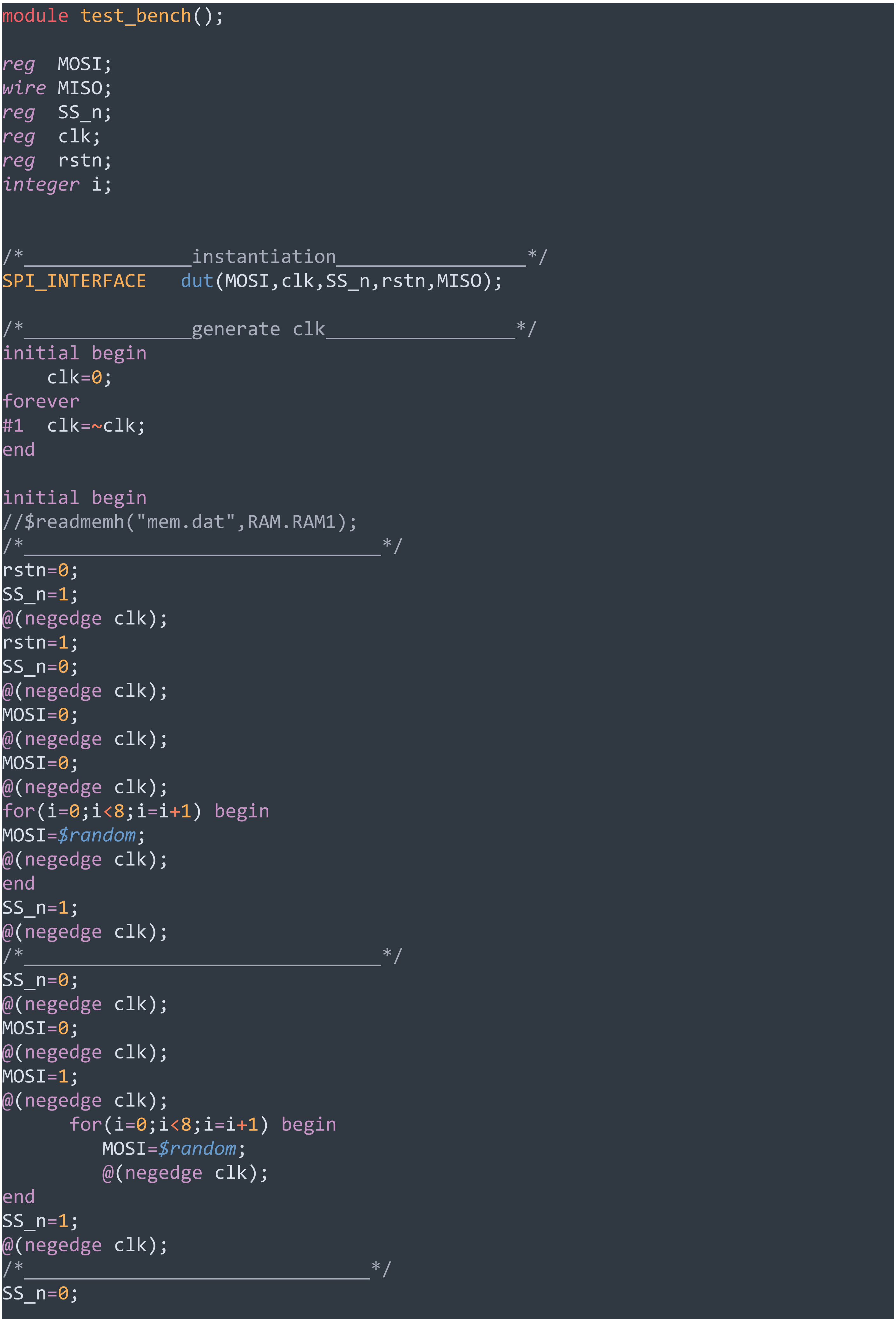


```
READ_ADDR: begin
            if(SS_n==1) begin
            NS=IDLE;
            end
          end
READ_DATA: begin

           if(SS_n==1) begin
           NS=IDLE;
           end
          end
 endcase
end
/*----------------state memory----------------*/
always@(posedge clk or negedge rstn) begin
    if(~rstn)
    CS=IDLE;
    else
    CS=NS;
end
/*----------------output logic----------------*/
always@(posedge clk) begin
    if(CS==WRITE )  begin
        rx_data[i]<=MOSI;
        i<=i-1;
               rx_valid<=1;
      end

    else if(CS==READ_ADDR) begin
        rx_data[j]<=MOSI;
         j<=j-1;
              rx_valid<=1;
              flag_read_addr<=1;
        end
    else if(CS==READ_DATA) begin
         rx_data[k]<=MOSI;
         k<=k-1;
         rx_valid<=1;
    end
    else rx_valid=0;
end

/*----------------output logic----------------*/
always@(posedge clk) begin
   if(tx_valid==1) begin
             MISO<=tx_data[l];
              l=l-1;
               end

    else MISO=0;
end
endmodule
```



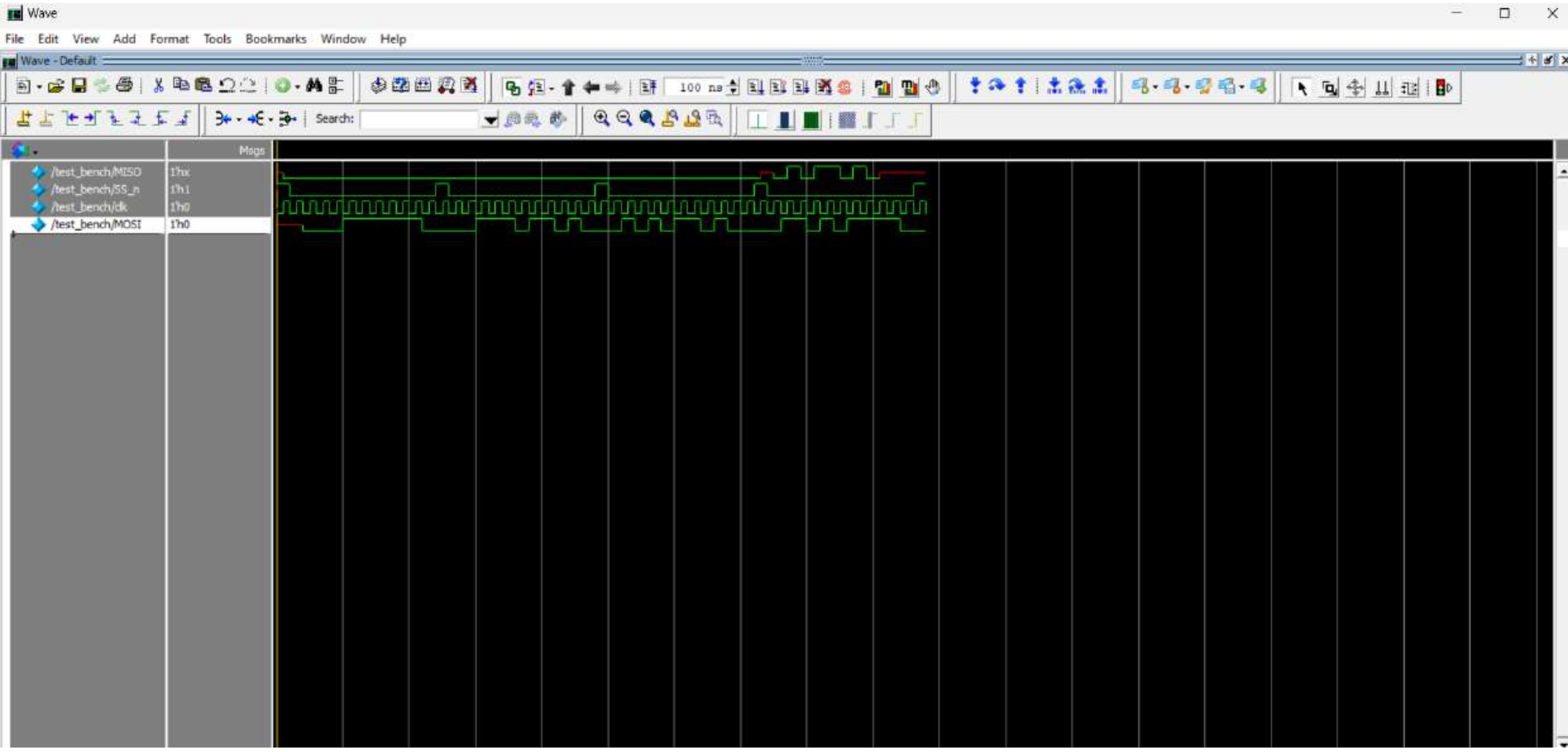
```
module RAM(clk,rstn,rx_valid,tx_valid,tx_data,rx_data);
parameter MEM_DEPTH=256;
parameter MEM_WIDTH=8;
input [9:0] rx_data;
input   clk,rstn,rx_valid;
output  reg [MEM_WIDTH-1:0] tx_data;
output  reg tx_valid;
 reg [7:0] din;
 reg [7:0] addr;

reg  [MEM_WIDTH-1:0] RAM [MEM_DEPTH-1:0];

always @(posedge clk) begin

if(rx_valid==1) begin

    if(rx_data[9:8]==2'b00)  begin
         addr<=rx_data[7:0];

    end

    else if(rx_data[9:8]==2'b01) begin
         din<=rx_data[7:0];

    end
    else if(rx_data[9:8]==2'b10) begin
          addr<=rx_data[7:0];

    end
    else if(rx_data[9:8]==2'b11) begin
          tx_valid=1;

   end
end
end
//RAM BLOCK;
always@(posedge clk or negedge rstn) begin
    if(~rstn)
      tx_data=0;
    else begin
                if(rx_valid==1)
                    RAM[addr]<=din;
                else if(tx_valid==1)
                    tx_data<=RAM[addr];

    end

end
endmodule
```

```verilog
module SPI_INTERFACE(MOSI,SCK,SS_n,rstn,MISO);
input  MOSI;
input  SCK;
input  SS_n;
input  rstn;
output MISO;

wire [7:0] tx_data;
wire tx_valid;
wire rx_valid;
wire [9:0] rx_data;


SPI_SLAVE #(3'b000,3'b001,3'b010,3'b011,3'b100)
SLAVE (SCK,rstn,MOSI,SS_n,tx_data,tx_valid,MISO,rx_valid,rx_data);


RAM  #(256,8) RAM1(SCK,rstn,rx_valid,tx_valid,tx_data,rx_data);



endmodule

//XC7A35TICPG236-1L
```

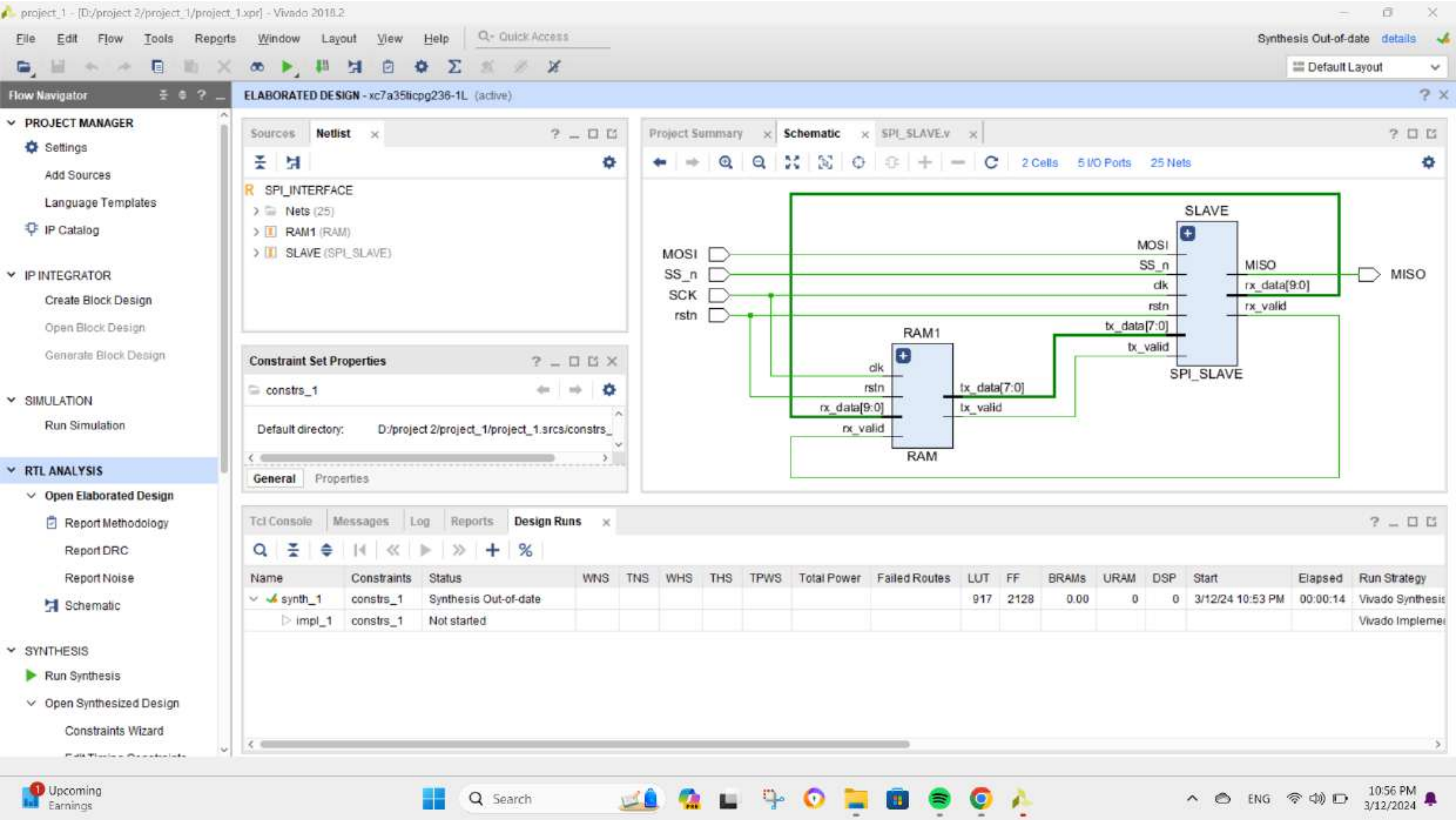


```verilog
module test_bench();

reg  MOSI;
wire MISO;
reg  SS_n;
reg  clk;
reg  rstn;
integer i;


/*_______________instantiation_________________*/
SPI_INTERFACE   dut(MOSI,clk,SS_n,rstn,MISO);

/*_______________generate clk_________________*/
initial begin
    clk=0;
forever
#1  clk=~clk;
end

initial begin
//$readmemh("mem.dat",RAM.RAM1);
/*________________________________*/
rstn=0;
SS_n=1;
@(negedge clk);
rstn=1;
SS_n=0;
@(negedge clk);
MOSI=0;
@(negedge clk);
MOSI=0;
@(negedge clk);
for(i=0;i<8;i=i+1) begin
MOSI=$random;
@(negedge clk);
end
SS_n=1;
@(negedge clk);
/*________________________________*/
SS_n=0;
@(negedge clk);
MOSI=0;
@(negedge clk);
MOSI=1;
@(negedge clk);
      for(i=0;i<8;i=i+1) begin
         MOSI=$random;
         @(negedge clk);
end
SS_n=1;
@(negedge clk);
/*_______________________________*/
SS_n=0;
```

```
@(negedge clk);
MOSI=1;
@(negedge clk);
MOSI=0;
@(negedge clk);
      for(i=0;i<8;i=i+1) begin
          MOSI=$random;
          @(negedge clk);
end
SS_n=1;
@(negedge clk);

/*________________________________*/
SS_n=0;
@(negedge clk);
MOSI=1;
@(negedge clk);
MOSI=1;
@(negedge clk);
     for(i=0;i<8;i=i+1) begin
           MOSI=$random;
           @(negedge clk);
end
SS_n=1;
@(negedge clk);
$stop;
end

endmodule
```



	Msgs
/test_bench/MISO	1'hx
/test_bench/SS_n	1'h1
/test_bench/clk	1'h0
/test_bench/MOSI	1'h0



project_1 - [D:/project 2/project_1/project_1.xpr] - Vivado 2018.2

Synthesis Out-of-date details

ELABORATED DESIGN - xc7a35ticpg236-1L (active)

SPI_INTERFACE
- Nets (25)
- RAM1 (RAM)
- SLAVE (SPI_SLAVE)

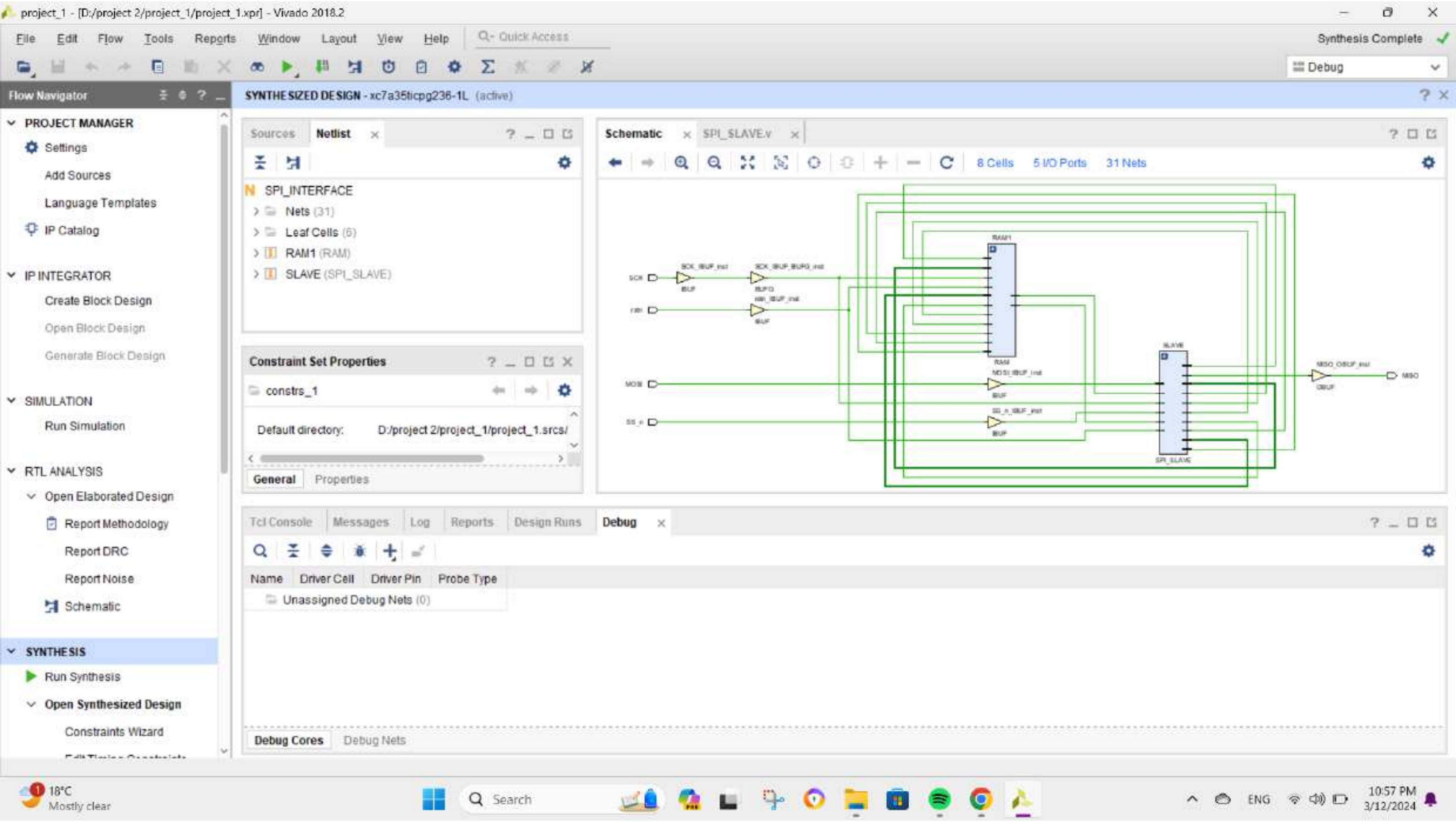
Constraint Set Properties

constrs_1

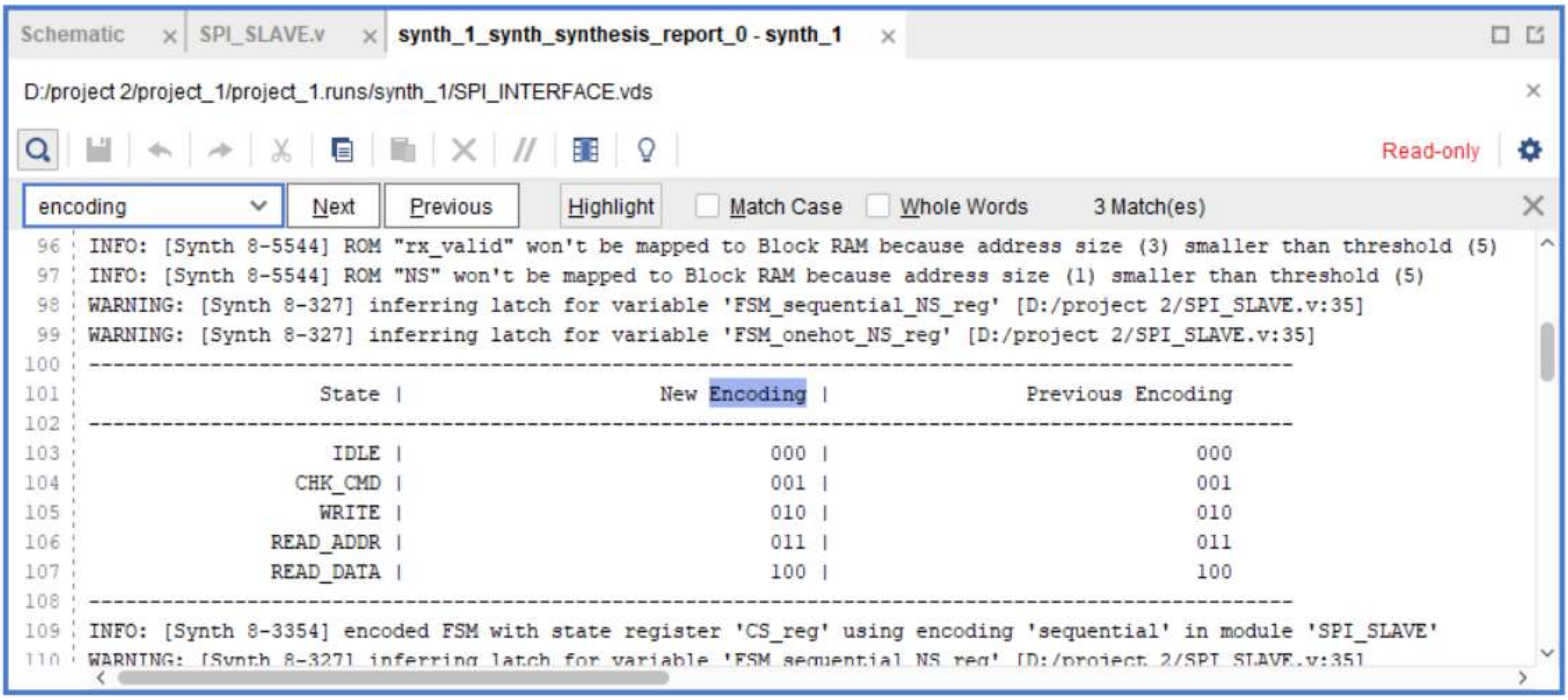
Default directory: D:/project 2/project_1/project_1.srcs/constrs_

2 Cells 5 I/O Ports 25 Nets

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed	Run Strategy
synth_1	constrs_1	Synthesis Out-of-date								917	2128	0.00	0	0	3/12/24 10:53 PM	00:00:14	Vivado Synthesis
impl_1	constrs_1	Not started															Vivado Impleme

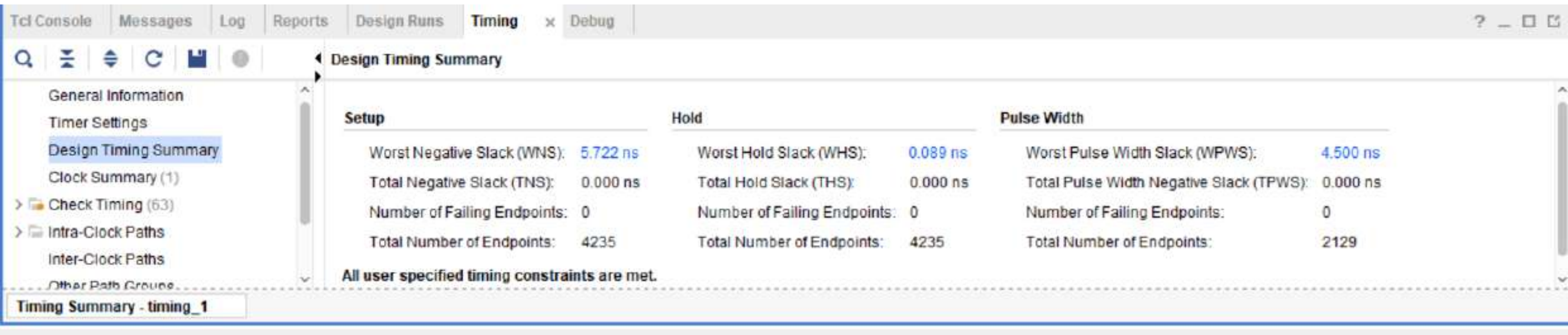


Default directory: D:/project 2/project_1/project_1.srcs/



D:/project 2/project_1/project_1.runs/synth_1/SPI_INTERFACE.vds

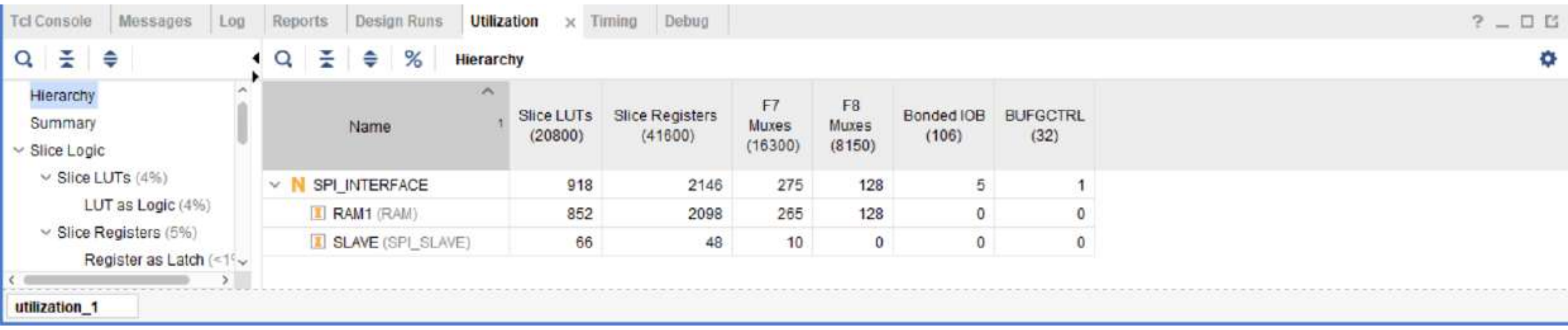
```
INFO: [Synth 8-5544] ROM "rx_valid" won't be mapped to Block RAM because address size (3) smaller than threshold (5)
INFO: [Synth 8-5544] ROM "NS" won't be mapped to Block RAM because address size (1) smaller than threshold (5)
WARNING: [Synth 8-327] inferring latch for variable 'FSM_sequential_NS_reg' [D:/project 2/SPI_SLAVE.v:35]
WARNING: [Synth 8-327] inferring latch for variable 'FSM_onehot_NS_reg' [D:/project 2/SPI_SLAVE.v:35]
---------------------------------------------------------------------------------------------------
                   State |                     New Encoding |                Previous Encoding
---------------------------------------------------------------------------------------------------
                    IDLE |                              000 |                              000
                 CHK_CMD |                              001 |                              001
                   WRITE |                              010 |                              010
               READ_ADDR |                              011 |                              011
               READ_DATA |                              100 |                              100
---------------------------------------------------------------------------------------------------
INFO: [Synth 8-3354] encoded FSM with state register 'CS_reg' using encoding 'sequential' in module 'SPI_SLAVE'
WARNING: [Synth 8-327] inferring latch for variable 'FSM_sequential_NS_reg' [D:/project 2/SPI_SLAVE.v:35]
```



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.722 ns	Worst Hold Slack (WHS):	0.089 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	4235	Total Number of Endpoints:	4235	Total Number of Endpoints:	2129

All user specified timing constraints are met.



Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	F8 Muxes (8150)	Bonded IOB (106)	BUFGCTRL (32)
SPI_INTERFACE	918	2146	275	128	5	1
RAM1 (RAM)	852	2098	265	128	0	0
SLAVE (SPI_SLAVE)	66	48	10	0	0	0